Московский государственный технический университет им. Н.Э.Баумана

Отчет по лабораторной работе № 3 по курсу “Разработка интернет-приложений”

«Python. Функциональные возможности.»

ИСПОЛНИТЕЛЬ:

студентка группы **ИУ5-53**

Паршева А. М.

_____

(подпись)

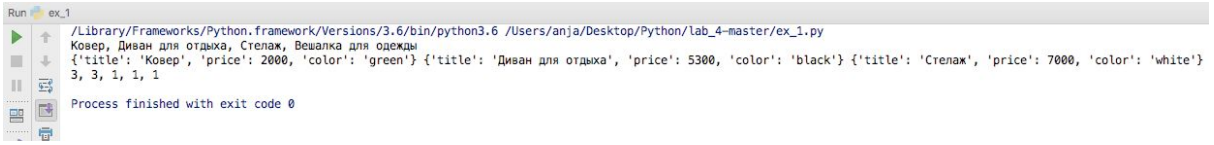
"__"_____2017 г.

1. Генератор.

```python
# Генератор вычленения полей из массива словарей
# Пример:
# goods = [
#    {'title': 'Ковер', 'price': 2000, 'color': 'green'},
#    {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}
# ]
# field(goods, 'title') должен выдавать 'Ковер', 'Диван для отдыха'
# field(goods, 'title', 'price') должен выдавать {'title': 'Ковер', 'price': 2000},

def field(items, *args):
    # проверка, если не пройдена- исключение
    assert len(args) > 0
    # Необходимо реализовать генератор
    # raise AssertionError

    if len(args) == 1:
        for elem in items:
            for key in args:
                a = elem.get(key)
                if a is not None:
                    yield a

    else:
        new_dict = {}

        for elem in items:
            for key in args:
                a = elem.get(key)
                new_dict[key] = a

            yield new_dict


# Генератор списка случайных чисел
# Пример:
# gen_random(1, 3, 5) должен выдать примерно 2, 2, 3, 2, 1
# Hint: реализация занимает 2 строки

def gen_random(begin, end, num_count):
    for num in range(num_count):
        yield random.randint(begin, end)
```

```python
#!/usr/bin/env python3
import ...


goods = [
    {'title': 'Ковер', 'price': 2000, 'color': 'green'},
    {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'},
    {'title': 'Стелаж', 'price': 7000, 'color': 'white'},
    {'title': 'Вешалка для одежды', 'price': 800, 'color': 'white'}
]

# Реализация задания 1
print(', ' .join(field(goods,'title')))

print(' '.join(map(str, field(goods, 'title', 'price','color'))))

print(', '.join(map(str, gen_random(1, 3, 5))))
```

Вывод:

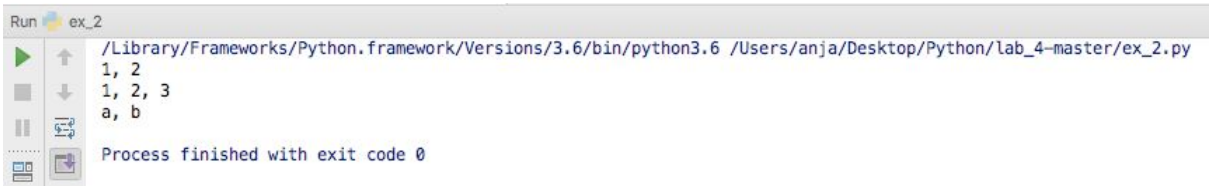


2. Итератор.

```python
# Итератор для удаления дубликатов
class Unique(object):
    def __init__(self, items, ignore_case=False,**kwargs):
        # Нужно реализовать конструктор
        # В качестве ключевого аргумента, конструктор должен принимать bool-параметр ignore_case,
        # в зависимости от значения которого будут считаться одинаковые строки в разном регистре
        # Например: ignore_case = True, Абв и АБВ разные строки
        #           ignore_case = False, Абв и АБВ одинаковые строки, одна из них удалится
        # По-умолчанию ignore_case = False
        self.items = items
        self.ignore_case = ignore_case

        if 'ignore_case' in kwargs:
            self.ignore_case = kwargs['ignore_case']

        self.returned = set()

    def __next__(self):
        # Нужно реализовать __next__

        for item in self.items:
            if type(item) == str and self.ignore_case == True:
                if item.lower() not in self.returned:
                    self.returned.add(item.lower())
                    return item
            else:
                if item not in self.returned:
                    self.returned.add(item)
                    return item
        raise StopIteration()

    def __iter__(self):
        return self
```

```python
#!/usr/bin/env python3
from librip.gens import gen_random
from librip.iterators import Unique

data1 = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
data2 = gen_random(1, 3, 10)
data3 = ['a', 'A', 'b', 'B']

# Реализация задания 2
print(', ' .join(map(str,Unique(data1))))

print(', ' .join(map(str,Unique(data2))))

print(', ' .join(map(str,Unique(data3, ignore_case=True))))
```

Вывод:



3. Сортировка массива.

```python
#!/usr/bin/env python3

data = [4, -30, 100, -100, 123, 1, 0, -1, -4]

# Реализация задания 3

print(sorted(data, key=lambda x: abs(x)))
```

Вывод:

4. Декоратор.

```python
def print_result(some_func):

    def decorated_func(*args,**kwargs):

        fun_list = some_func(*args,**kwargs)
        print(some_func.__name__)
        if type(fun_list) == list:
            print('\n'.join(map(str, fun_list)))

        elif type(fun_list) == dict:
            [print(key, '=', value) for key, value in fun_list.items()]

        else:
            print(fun_list)

        return fun_list


    return decorated_func

```

```python
from librip.decorators import print_result

# Необходимо верно реализовать print_result
# и задание будет выполнено

@print_result
def test_1():
    return 1


@print_result
def test_2():
    return 'iu'


@print_result
def test_3():
    return {'a': 1, 'b': 2}


@print_result
def test_4():
    return [1, 2]


test_1()
test_2()
test_3()
test_4()
```

Вывод:

5. Контекстный менеджер.

```python
import time

import contextlib


@contextlib.contextmanager
def timer():
    start = time.time()
    yield


    end = time.time()
    print(end - start)
```

```
import ...

with timer():
    sleep(5.5)

```

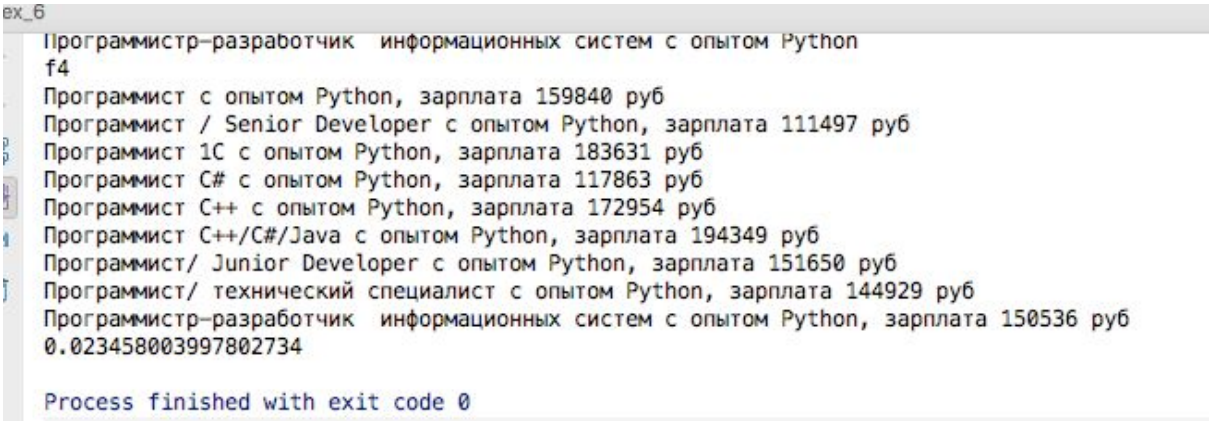
Вывод:

```
Run ex_5
/Library/Frameworks/Python.framework/Versions/3.6/bin/python3.6 /Users/anja/Desktop/Python/lab_4-master/ex_5.py
5.50401496887207

Process finished with exit code 0
```

6. Обработка данных.

```
import sys
from librip.ctxmngrs import timer
from librip.decorators import print_result
from librip.gens import field, gen_random
from librip.iterators import Unique as unique

path = None

# Здесь необходимо в переменную path получить
# путь до файла, который был передан при запуске

with open('data_light.json') as f:
    data = json.load(f)


# Далее необходимо реализовать все функции по заданию, заменив `raise NotImplemented`
# Важно!
# Функции с 1 по 3 должны быть реализованы в одну строку
# В реализации функции 4 может быть до 3 строк
# При этом строки должны быть не длиннее 80 символов

@print_result
def f1(arg):
    return list(sorted(unique(field(arg, 'job-name'), ignore_case=True)))

@print_result
def f2(arg):
    return list(filter(lambda line: line.startswith('Программист') or line.startswith('программист'), arg))


@print_result
def f3(arg):
    return list(map(lambda line: line + ' с опытом Python',arg))



@print_result
def f4(arg):
    random = list(gen_random(100000,200000,len(arg)))
    salary = list(map(lambda line:', зарплата ' + str(line) + ' руб', random))

    return list(map(lambda y: y[0] + y[1], zip(arg, salary)))


with timer():
    f4(f3(f2(f1(data))))
```

Вывод:


ex_6
Программистр-разработчик информационных систем с опытом Python
f4
Программист с опытом Python, зарплата 159840 руб
Программист / Senior Developer с опытом Python, зарплата 111497 руб
Программист 1С с опытом Python, зарплата 183631 руб
Программист C# с опытом Python, зарплата 117863 руб
Программист C++ с опытом Python, зарплата 172954 руб
Программист C++/C#/Java с опытом Python, зарплата 194349 руб
Программист/ Junior Developer с опытом Python, зарплата 151650 руб
Программист/ технический специалист с опытом Python, зарплата 144929 руб
Программистр-разработчик информационных систем с опытом Python, зарплата 150536 руб
0.023458003997802734
Process finished with exit code 0